Information Computer Systems Ltd. 2002 - 052

DESIGN NOTE / REPORT

TITLE SPECIFICATION OF THE ALP TYPE 3

AUTHOR D. ILLING.

DATE 8.3.73.

COMMENTS BY

CIRCULATION

INDEX

1.0. SUMMARY

The ALP 3 is the largest of the three Arithmetic and Logic Processors available in the Multum range. This document, in conjunction with 2002 - 042, defines its facilities. Since the ALP 3 is essentially the same as ALP 2, but with an extended instruction set, this specification deals only with those features which are peculiar to the ALP 3. The ALP 3 conforms in all respects to 2002 - 042, except where there may be any conflict with the information in this document.

2.0. RELATED DOCUMENTS

2002 - 042: Specification for the ALP Type 2.

2008 - 018: Design Note, "Floating Point Formats and Instructions".

3.0. GENERAL DESCRIPTION

The type 3 Arithmetic and Logic Processor is based upon the ALP 2, and has all the features of that processor plus hardware floating-point arithmetic facilities.

The ALP 3 has the same basic 16-bit word and register structure as ALP 2, and uses identical control logic for the non-floating-point instructions.

The floating-point instructions are based upon a 32-bit word, and use the double-length accumulator E (i.e. registers A and B in cascade) as the source of the basic operand and the destination of the instruction result. Those instructions involving a second operand use a double-length word from store, obtained in the same manner as for ordinary double-length functions.

The floating-point hardware - control logic, plus additional registers and arithmetic units - is incorporated on one logic card. An ALP 3 therefore consists physically of the seven ALP 2 logic cards (eight with a full control panel) plus the ALP 3 expansion card.

4.0. INTERNAL ORGANISATION

The register structure, etc. of ALP 3 is identical to the ALP 2 except for the additional floating-point hardware. Only the latter, therefore, need be dealt with here.

A floating-point word consists of two parts, the exponent and the mantissa (see section 5.0.). There are two separate sets of registers to deal with these. The diagram at fig. 1 shows the floating-point register structure.

4.1. The Mantissa Registers

4.1.1. Register AM - This is a 23-bit register used as the accumulator for the mantissa operations. It is initially loaded from bits 0-23 of register E.

4.1.2. Register IM - This is a 23-bit register into which the mantissa part of a memory-held operand is loaded.

4.1.3. Register G - This is a 23-bit general purpose register used in operations on the mantissae.

4.1.4. The Mantissa Arithmetic Unit (MAU) - This unit provides the ability to add, subtract, negate, test and increment the 23-bit mantissa parts of the floating point words.

4.2. The Exponent Registers

4.2.1. Register AE - This is an 8-bit register used as the accumulator for exponent operations. It is initially loaded from bits 24-31 of register E.

4.2.2. Register IE - This is an 8-bit register into which the exponent part of a memory-held operand is loaded.

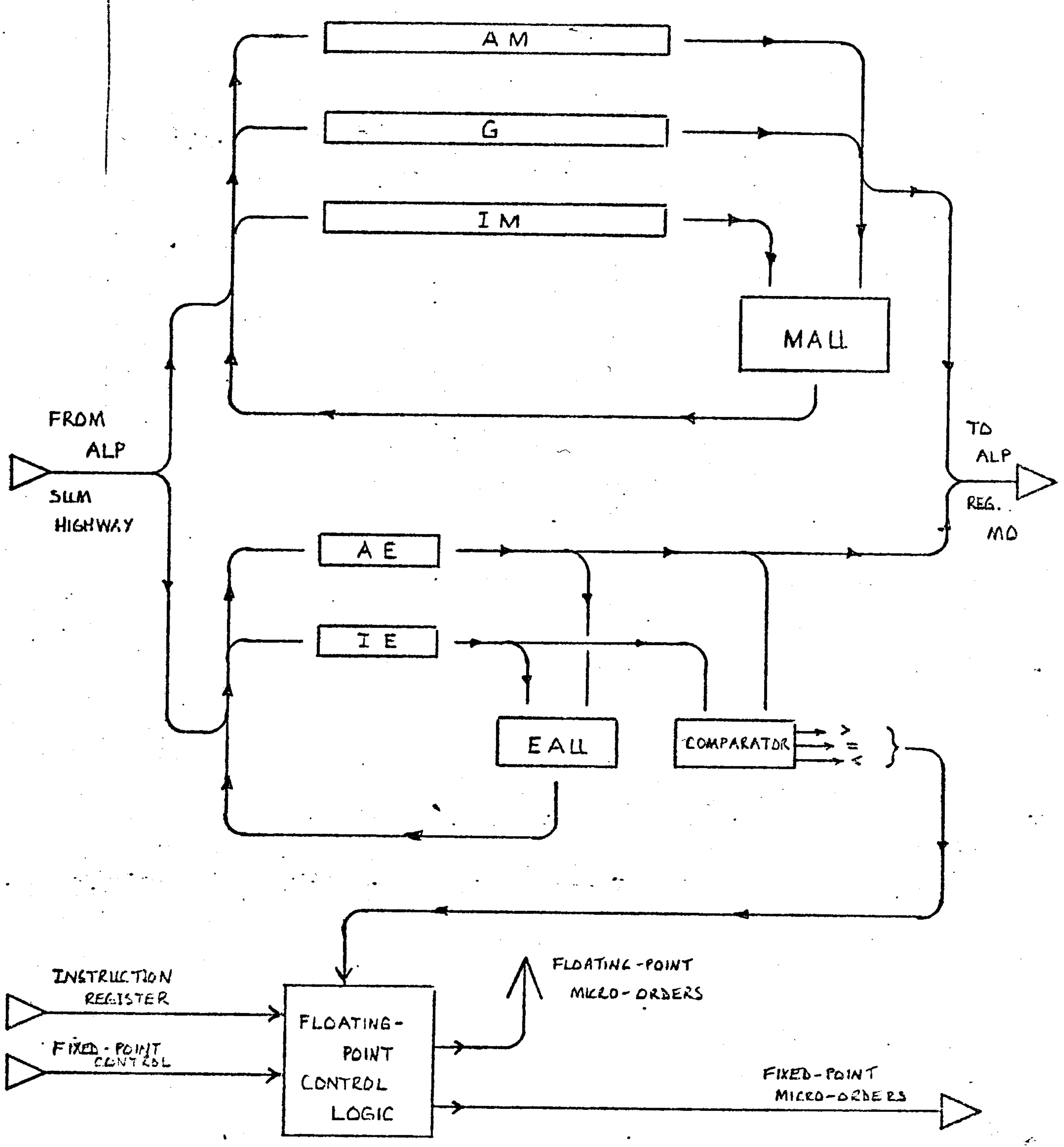
4.2.3. The Exponent Arithmetic Unit (EAU) - This unit provides the ability to add or subtract AE and IE.

4.2.4. The Exponent Comparator - This examines AE and IE to determine for the control logic which is the greater.

4.3. Floating Point Control and Data Manipulation

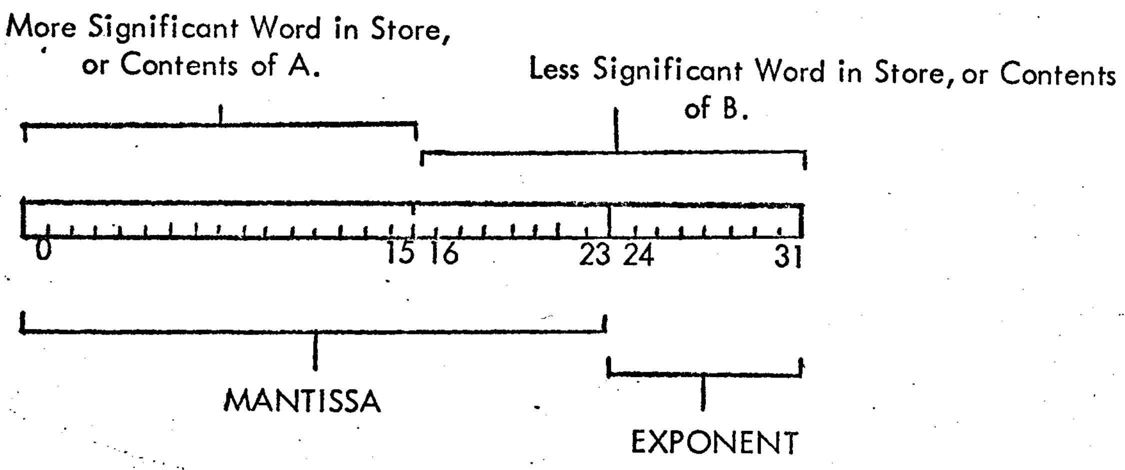
The floating-point control logic is able to generate micro-orders to manipulate the main register structure as well as the floating-point registers. The normal Group 1 control logic is used to calculate the effective address when there is a second operand. The operands enter the floating-point registers from the SUM highway, and the final result of a floating-point instruction re-enters the main register structure via the register MO.

FIGURE 1 - FLOATING-POINT HARDWARE STRUCTURE



5.0. FLOATING-POINT DATA FORMAT

The floating-point instructions operate on a 32-bit word having the format described below. This word is held as two consecutive 16-bit words in store or in the double-length accumulator, E (i.e. registers A and B).


More Significant Word in Store, or Contents of A.
Less Significant Word in Store, or Contents of B.
0
15
16
23
24
31
MANTISSA
EXPONENT

5.1. The Mantissa (M)

The Mantissa is the value part of a floating point number. It is represented as a signed binary fraction. On ALP 3 the mantissa is a 24-bit signed 2's compliment fraction, and will normally be in the following range (see 5.3.):-

M positive	$\frac{1}{2} \leqslant M \leqslant (1 - 2^{-23})$
M negative	$-1 \leqslant M \leqslant -(\frac{1}{2}+2^{-23})$

5.2. The Exponent

The exponent part of a floating-point number specifies the position of the binary point; or in other words is a multiplier, expressed as a power of 2, acting on the mantissa. On ALP 3 the exponent is held as an 8-bit number in excess-128 form - i.e. the values -128 to +127 are held as 0 to 255.

5.3. Standardisation

In order to maintain the greatest number of significant digits at any given time, and hence ensure the greatest precision, a floating-point number is standardised after every floating-point instruction.

The procedure for standardisation is to examine the two most significant bits of the mantissa. If they are not identical, then the number is standardised. If they are identical then the mantissa is shifted left one place and the exponent is decremented. This is repeated until the condition for standardisation is reached (i.e. $M\emptyset \neq M1$), or an underflow occurs (see 6.3.2.).

Thus a standardised number has a mantissa lying in the range:-

M positive	$\frac{1}{2} \leqslant M \leqslant (1 - 2^{-23})$
M negative	$-1 \leqslant M \leqslant -(\frac{1}{2} + 2^{-23})$

If floating-point numbers are operated on in a non standardised state, inaccurate or unrelated results may be produced.

Where possible, guard-digits are preserved during floating-point operations so that in the final standardisation, if shifting is required, they can be re-inserted at the least-significant end of the mantissa to maintain precision.

5.4. Range and Precision

The range of values which can be held in the format described above, with its constraints, is as follows:-

Positive numbers: $\frac{1}{2} \times 2^{-128} \leqslant N \leqslant (1-2^{-23}) \times 2^{127}$

Negative numbers: $-(\frac{1}{2}+2^{-23}) \times 2^{-128} \leqslant N \leqslant -1 \times 2^{127}$

Note that the positive and negative ranges are not symmetrical. To a reasonable approximation:

$$2^{-129} \leqslant |N| \leqslant 2^{127} ,$$

or to base 10:

$$1.4 \times 10^{-39} \leqslant |N| \leqslant 1.7 \times 10^{38}$$

The precision is 23 bits, or approximately 7 decimal digits.

5.5. The Value Zero

The value zero is held as a mantissa of zero and zero exponent. Note that this is an exception to the standardised form specified above.

6.0. INSTRUCTION SET

The ALP 3 instruction set includes all those instructions and formats available on the ALP 2, and in addition the floating-point instructions described below. (On ALP 2 any of these instructions causes a Function Trap Interrupt.) The common instructions are fully described in 2002 - 42 and will not be dealt with here.

6.1. Group 1 Floating Point Instructions

6.1.1. Literal Format

When the instruction is Format 1 Mode 0, the 8-bit "D" field is used as data. It is taken as an unsigned binary integer, and is converted to a positive floating-point number before being used as the second operand in the instruction. (see 2002-042 section 8.1.1.).

6.1.2. Non-Literal Formats

As with other Group 1 instructions, a memory-held operand is used. The Effective Address from which this operand is extracted is determined by the same procedure as for other Group 1 instructions (see 2002-042 sections 8.1.1. to 8.1.4. inclusive). The contents of the Effective Address are assumed to be a standardised floating-point number.

6.1.3. Address Scaling

Since floating-point instructions are in effect double-word functions, the Effective Address will be scaled where a hardware register is the final displacement - i.e. in Format 3 Modes 2-5 and Format 4. (See 2002-042 section 8.1.5.)

6.1.4. The Functions

6.1.4.1. ADDF : $(OC)_{16}$ A copy of the floating-point number defined by the effective address is added to the floating-point number contained in E. The result is standardised and overwrites the original contents of E.

$$(E_f) := (E_f) + (Q_f)$$

Where the suffix f denotes a standardised floating-point number.

6.1.4.2. SUBF : $(OD)_{16}$ A copy of the floating-point number defined by the effective address is subtracted from the floating-point number contained in E. The result is standardised, and overwrites the original contents of E. The subtraction is effected by a process of negating and adding. It is therefore not possible to subtract the largest negative number in the range because an overflow condition occurs during the negation stage (see 6.3.1.)

$$(E_f) := (E_f) - (Q_f)$$

6.1.4.3. MLTF : $(OE)_{16}$

A copy of the floating-point number defined by the effective address is multiplied by the floating-point number contained in E. The result is standardised and overwrites the original contents of E. (Where the result of multiplying the mantissae is greater than 24 bits, those bits of significance less than 2^{-23} after standardisation are lost. There is no rounding.) When the mantissa of the floating-point number in the effective address has the maximum negative value (i.e. -1), that number is destandardised before the multiplication by an arithmetic right-shift and an increment of its exponent. It is therefore not possible to multiply by the largest negative number in the range because an overflow condition occurs at this point (see 6.3.1.).

$$(E_f) := (E_f) \times (Q_f)$$

6.1.4.4. DIVF $(OF)_{16}$

The floating-point number contained in E is divided by a copy of the floating-point number defined by the effective address. The result is standardised and overwrites the original contents of E. Before the division, the contents of the effective address are tested, and if zero a Zero Divide condition is detected the instruction is abandoned (see 6.3.4.). It is possible for the division of the mantissae to yield a non-fractional result.

Before the division, therefore, the relative magnitudes of the mantissae are examined. If the magnitude of the mantissa of the dividend is greater than or equal to that of the divisor, the dividend is destandardised by an arithmetic right-shift and an increment of the exponent. This may lead to an overflow condition, in which case the instruction is abandoned. (see 6.3.1.).

$$(E_f) := (E_f) \div (Q_f)$$

6.2. Group 5 Floating-Point Instruction

There are six instructions in Group 5 which are floating-point functions. They are non-privileged.

6.2.1. NEGF · $(1A)_{16}$

The floating-point number contained in E is negated, standardised if necessary, and the result overwrites the original contents of E. Since the positive and negative ranges of a floating point number are not the same, an underflow or an overflow may occur (see 6.3.1. and 6.3.2.).

$$(E_f); = - (E_f)$$

6.2.2. FLTI : $(1B)_{16}$

The contents of E are taken to be a fixed-point integer, with the binary point to the right of the least significant bit. This integer is converted to a standardised floating-point number which overwrites the original contents of E.

6.2.3. FLTF : $(1C)_{16}$	The contents of E are taken to be a fixed-point fraction, with the binary point to the right of the most significant bit. This fraction is converted to a standardised floating-point number which overwrites the original contents of E.
6.2.4. FIXI : $(1D)_{16}$	The contents of E are taken to be a standardised floating-point number. If the exponent is greater than 31 (i.e. 159 as held), the number cannot be expressed as a 32-bit integer; this is a No Fix failure (see 6.3.3.). Provided that the exponent is not greater than 31, the integer part of the number, expressed as a normal fixed-point double-length integer, overwrites the original contents of E. Any fractional part of the number is lost. Where the number is wholly fractional, E becomes zero.
6.2.5. FIXF : $(1E)_{16}$	The contents of E are taken to be a stardardised floating-point number. If the number has an integer part, it cannot be fixed as a fraction; this is a No Fix failure (see 6.3.3.). Provided that the number is wholly fractional, it is converted to a fixed-point double-length fraction which overwrites the original contents of E. If the magnitude of the number is less than 2^{-31}, E becomes zero.

6.2.6. STND : $(1F)_{16}$

The contents of E are taken to be a floating-point number, which may or may not be standardised. This number is put through the standardisation procedure (see 5.3.) and the result overwrites the original contents of E. It is possible for an Underflow to occur during this operation (see 6.3.2.)

6.3. Floating-Point Failures

There are four types of failure which can occur during a floating-point instruction (besides any normal failures, such as a Violation occurring during accessing a memory-held operand). These are detailed below. When any of these is detected during the course of an instruction the instruction is abandoned, and an internal interrupt type 4 ("Violation") is set, together with bit 6 of the Conditions Register and a code in bits 14 and 15 (see 7.1. and 7.2.). The original contents of the double-length accumulator, E, are undisturbed.

6.3.1. Floating Overflow - An overflow occurs if the exponent part of a floating-point number becomes too large to be accomodated (i.e. exceeds the value +127 ; 255 as held). It can occur in any of the following cases:-

a) In an ADDF or SUBF instruction if the result lies outside the range of standardised floating-point numbers.

b) As a result of trying to negate the largest negative number in the range. Note that this can occur not only in a NEGF instruction, but also in SUBF if the subtrahend is that number (see 6.1.4.2.).

c) In a MLTF instruction if the store-held multiplicand is the largest negative number in the range (see 6.1.4.3.)

d) In a DIVF instruction if destandardisation of the dividend is required (see 6.1.4.4.) and this results in an exponent greater than +127.

e) In a MLTF or DIVF instruction if the result of respectively adding or subtracting the exponents is greater than +127.

6.3.2. Floating Underflow - An underflow occurs if the exponent-part of a floating-point number becomes too small to be accomodated (i.e. becomes less than -128 ; zero as held). It can occur in any of the following cases:-

a) During the standardisation procedure (see 5.3.), if number is not standardised when the exponent reaches -128.

b) In a MLTF or DIVF instruction if the result of respectively adding or subtracting the exponents is less than -128.

6.3.3. No Fix Failure - This can occur in either a FIXI or FIXF instruction. In the former case there is No Fix if the integer part of the floating-point number in E is too large to be expressed as a 32-bit two's-compliment fixed-point integer. In the latter case there is No Fix if the number is not wholly fractional.

6.3.4. Zero Divide - This occurs in a DIVF instruction if the divisor is zero.

7.0: CONDITIONS REGISTER

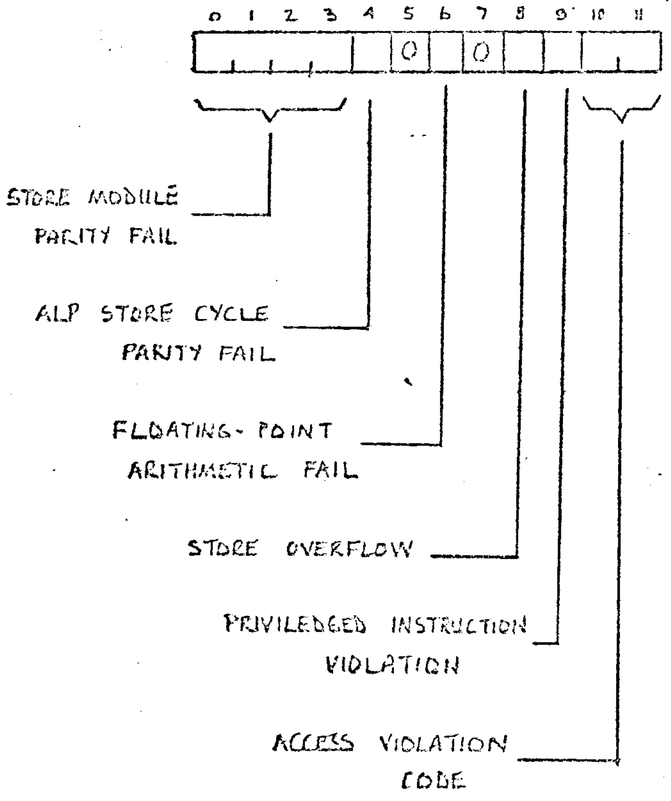
The ALP 3 Conditions Register is identical to that of ALP 2 (see 2002-042 section 9.0) with the exception of the following:-

7.1. Additional Machine Conditions Bits

In addition to those bits in the Machine Conditions field which are used on an ALP 2, on ALP 3 bit 6 is also used.

CR 6 is set whenever a Floating-point Arithmetic failure is detected.

The machine conditions field of an ALP 3 is therefore thus:-


0 1 2 3 4 5 6 7 8 9 10 11
0
0
STORE MODULE PARITY FAIL
ALP STORE CYCLE PARITY FAIL
FLOATING-POINT ARITHMETIC FAIL
STORE OVERFLOW
PRIVILEDGED INSTRUCTION VIOLATION
ACCESS VIOLATION CODE

7.2. Additional Program Conditions

The bits 12 and 13 of the conditions register have the same meaning as an ALP 2. Bits 14 and 15, however are redefined, and are called the Arithmetic Failure Indicator.

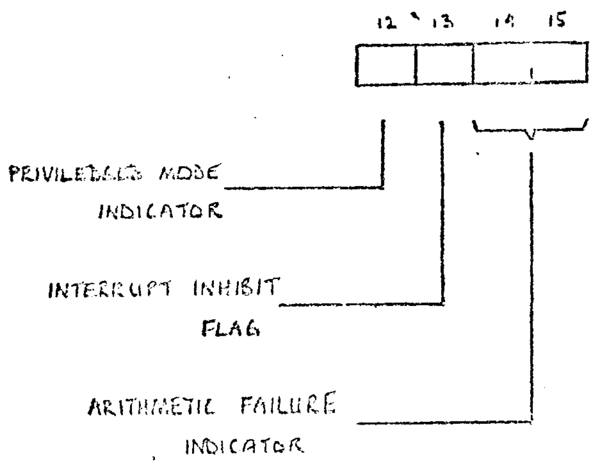
The interpretation of the Arithmetic Failure Indicator is dependent on the state of bit 6 of the conditions register.

If bit 6 is not set, the indicator bits show when fixed-point Arithmetic Overflow and Carry Overflow have occurred, exactly as defined in 2002-042 section 9. 2.

If bit 6 is set, the contents of the Indicator show which type of floating-point failure has occurred, as follows:-

CR 14	CR 15			
0	0	-	No Fix	(see 6.3.3.)
0	1	-	Overflow	(see 6.3.1.)
1	0	-	Underflow	(see 6.3.2.
1	1	-	Zero Divide	(see 6.3.4.)

The Program Conditions Field of an ALP 3 therefore thus:- "A"



It is important to note that:-

a) The rules for setting and resetting conditions register bits 14 and 15 as specified in 2002-042 still apply.

b) Any given arithmetic failure only sets appropriate bits, it does not completely re-write the contents of the Arithmetic Failure Indicator. Therefore if, for example, a fixed point Arithmetic Overflow has occurred prior to a Floating Overflow, without CR14 having been cleared in the meantime, both bits of the Indicator will be set. It is therefore important that the Arithmetic Failure Indicator be cleared as soon as its contents are no longer needed.

8.0. INTERRUPTS

Interrupts on the ALP 3 are exactly as specified for ALP 2 (see 2002-042 section 10.), except that there is an additional cause for setting the internal interrupt type 4 ("Violation").

A "Violation" interrupt is set whenever a floating point failure is detected (see 6.3.).

APPENDIX A

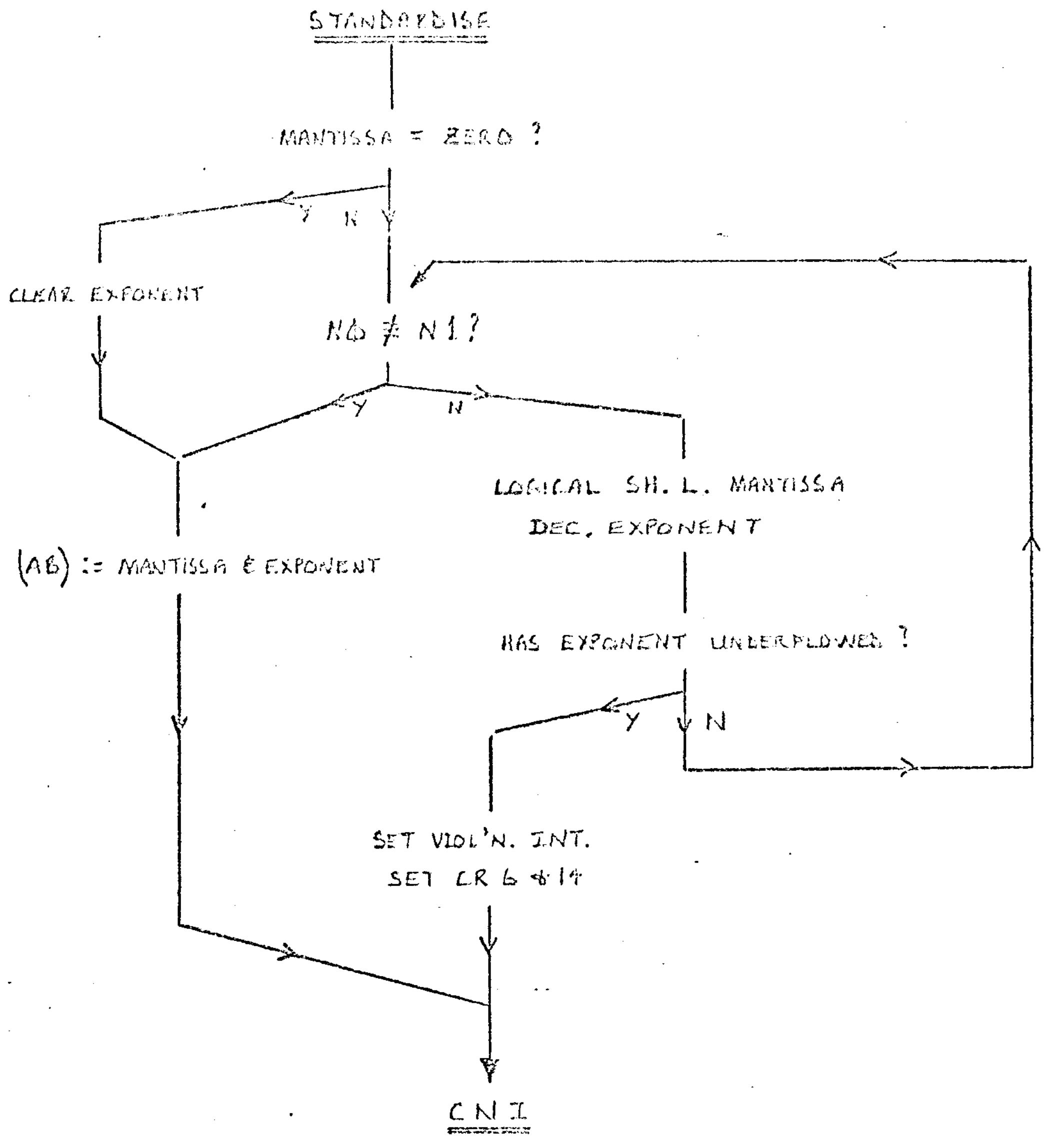
The following flowcharts describe the procedures used to impliment the ALP 3 Floating Point functions.

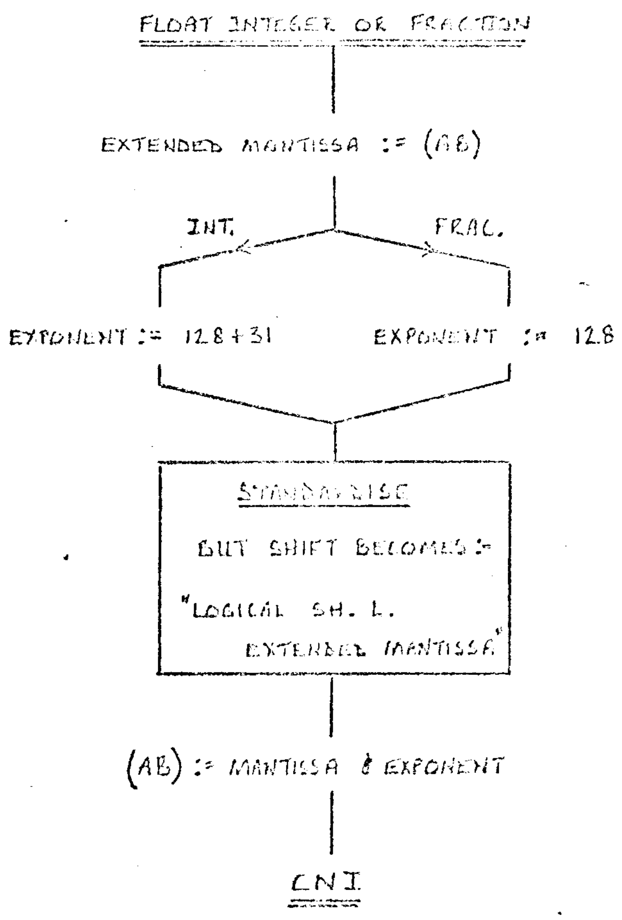
A1 - The Standardise procedure, as used at the end of all instructions and in the Group 5 instruction STND. Note that in post-instruction standardisation, the mantissa is sometimes longer than 24 bits. When standardisation is complete the mantissa is truncated by the right of the 24th bit.

A2 - The Fix and Float procedures which translate numbers between floating and fixed-point formats. (The Group 5 instructions FLTI, FLTF, FIXI, FIXF).

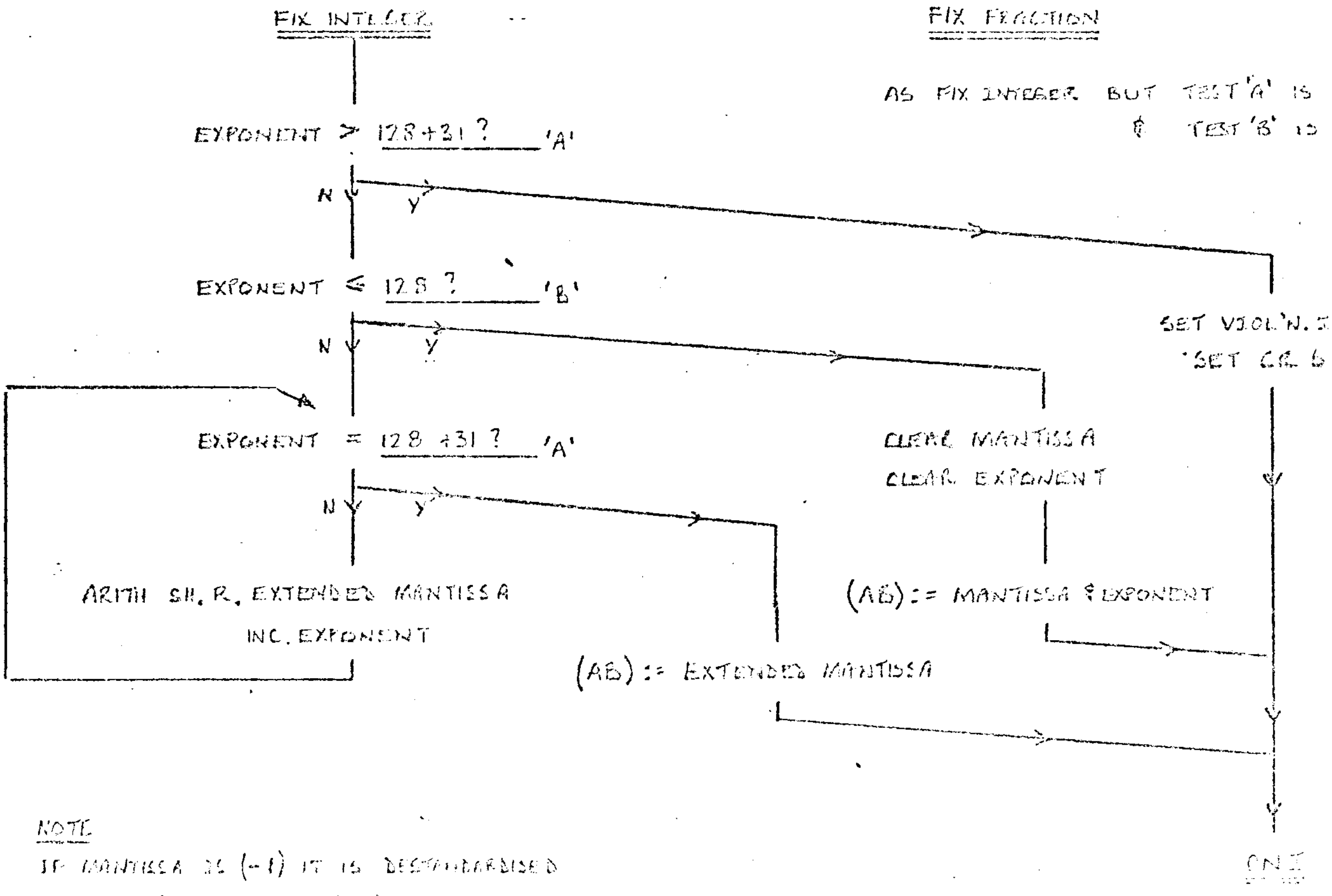
A3 - The Negate procedure used in the Group 5 instructions NEGF.

A4-A6 - The Arithmetic procedures used for the Group 1 instructions. (ADDF, SUBF, MLTF, DIVF).


STANDARDISE
MANTISSA = ZERO ?
Y
N
CLEAR EXPONENT
N0 ≠ N1 ?
Y
N
LOGICAL SH. L. MANTISSA
DEC. EXPONENT
HAS EXPONENT UNDERFLOWED ?
Y
N
(AB) := MANTISSA & EXPONENT
SET VIOL'N. INT.
SET CR 6 & 14
CNI



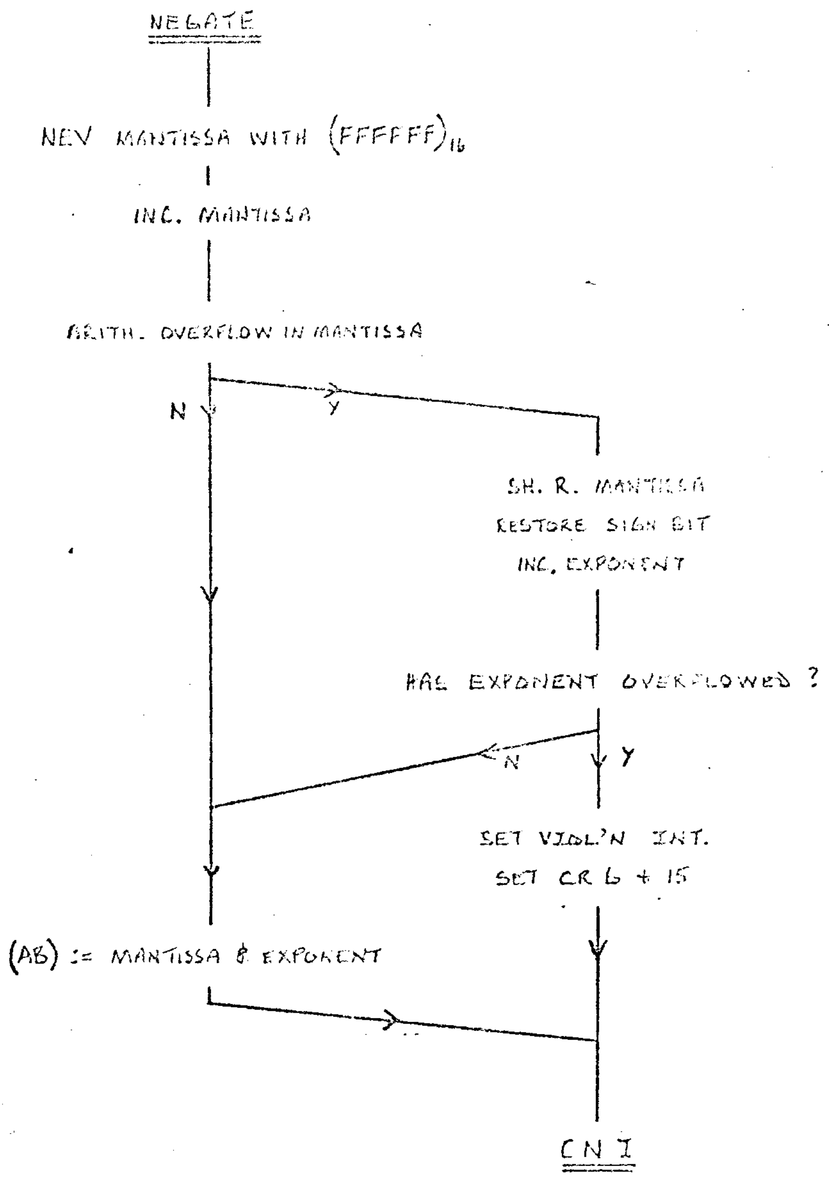
NOTE
"EXTENDED MANTISSA" IS
MANTISSA WITH EXTRA 8 BITS
ADDED TO LEAST SIGNIF. END

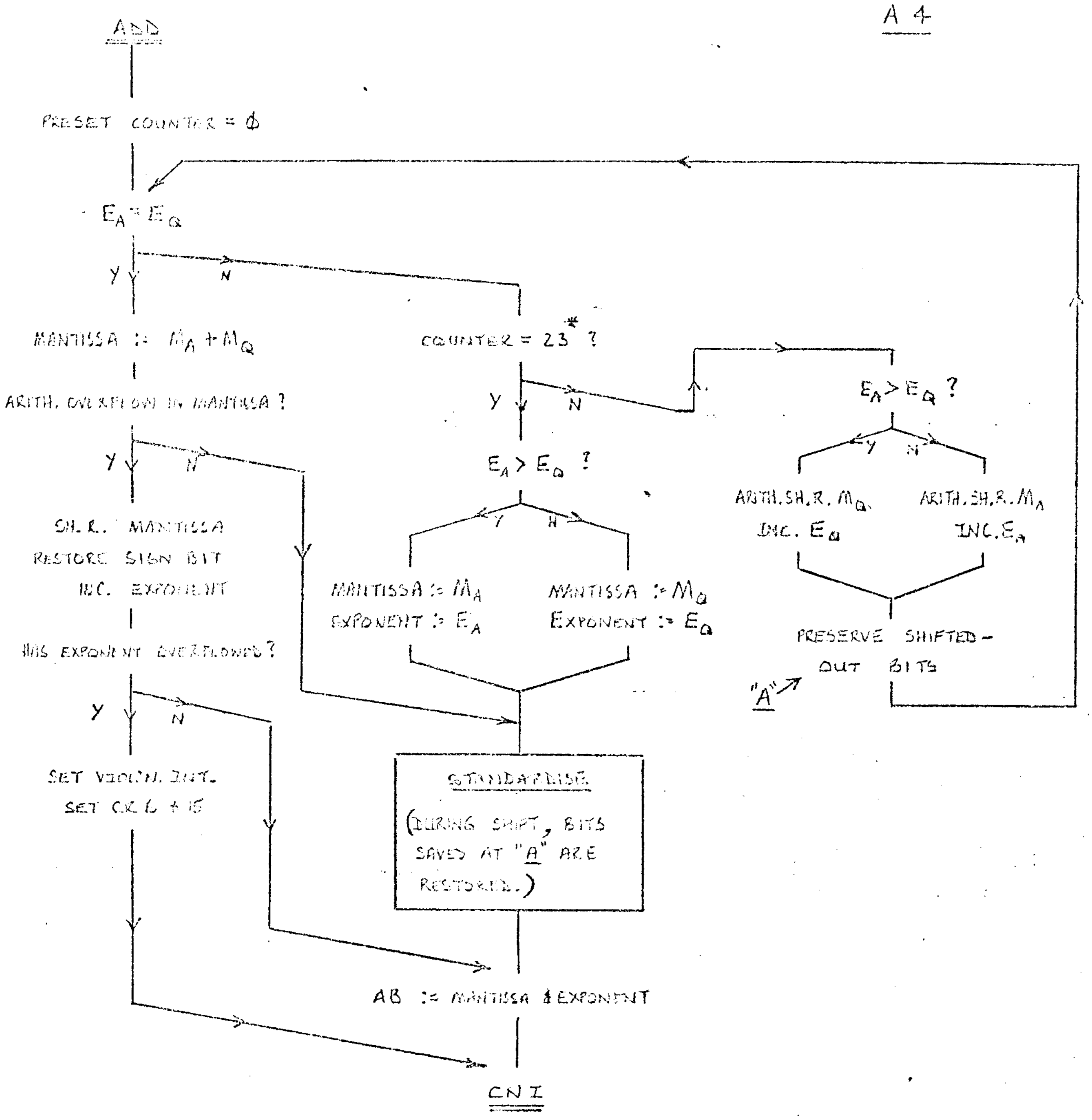


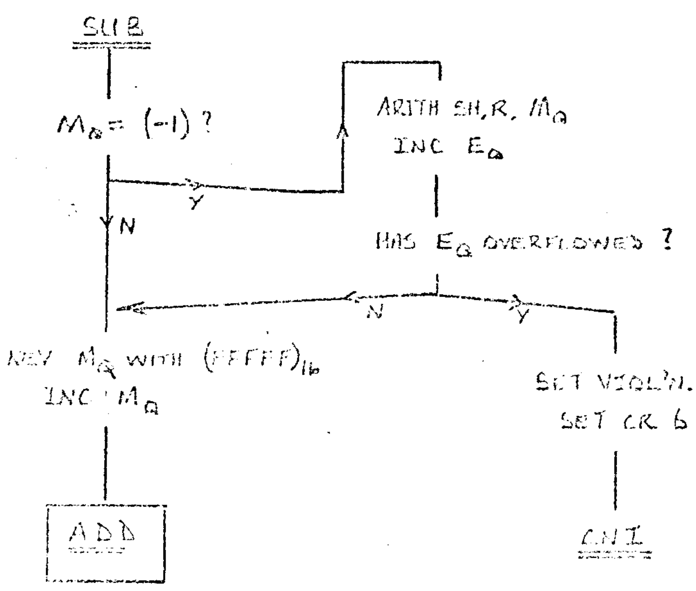
FIX FRACTION

AS FIX INTEGER BUT TEST 'A' IS 128
& TEST 'B' IS 128-

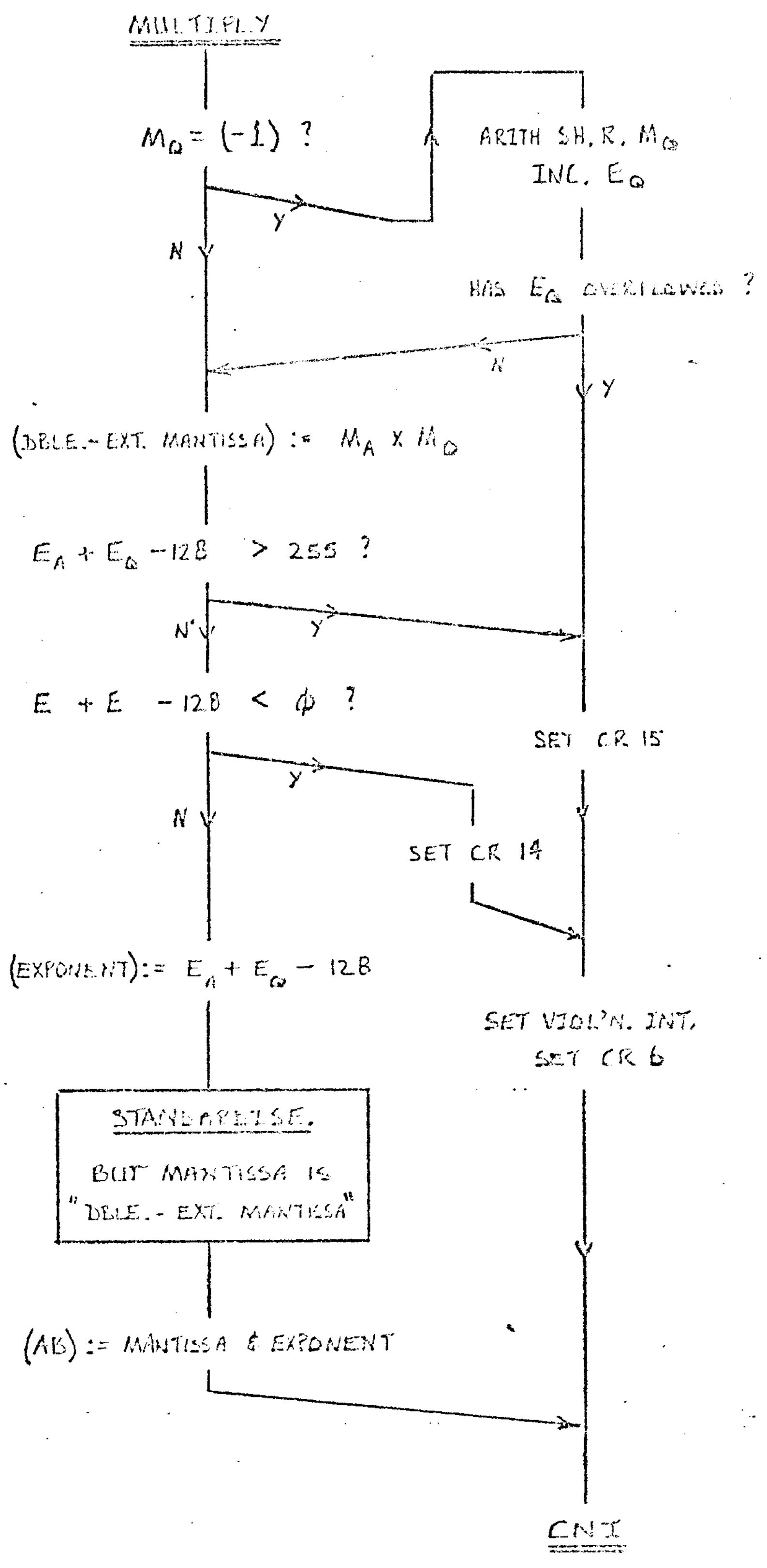
NOTE
IF MANTISSA IS (-1) IT IS DESTANDARDISED
BEFORE A FIX INSTRUCTION.


NEGATE
NEV MANTISSA WITH $(FFFFFF)_{16}$
INC. MANTISSA
ARITH. OVERFLOW IN MANTISSA
N
Y
SH. R. MANTISSA
RESTORE SIGN BIT
INC. EXPONENT
HAS EXPONENT OVERFLOWED ?
N
Y
SET VIOL'N INT.
SET CR 6 + 15
(AB) := MANTISSA & EXPONENT
CNI

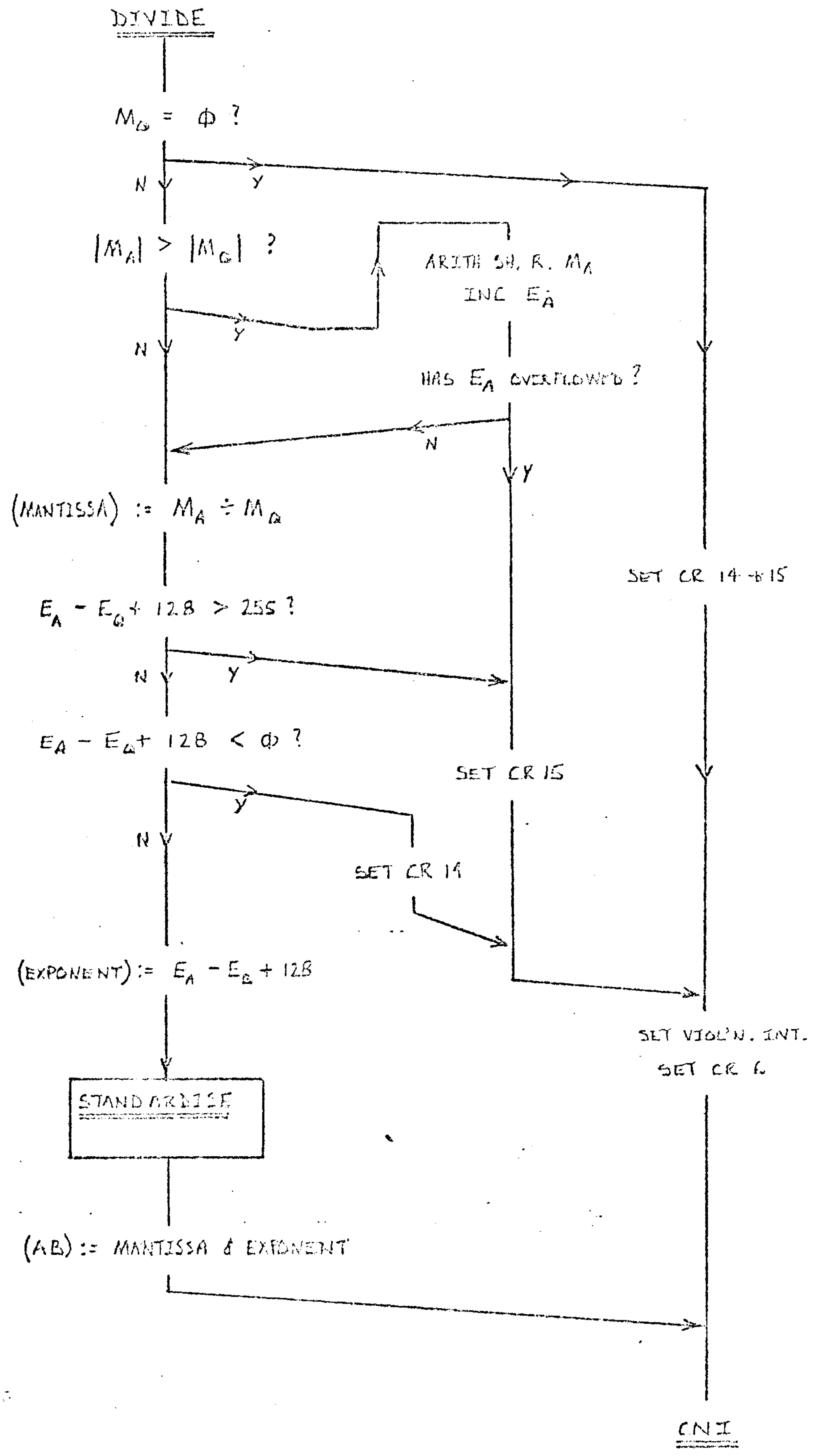




* IF MANTISSA BEING SHIFTED (I.E. M_Q IF $E_A > E_Q$) IS NEGATIVE, "COUNTER = 24" USED.


MULTIPLY
$M_Q = (-1)$?
Y
N
ARITH SH. R. M_Q
INC. E_Q
HAS E_Q OVERFLOWED ?
N
Y
(DBLE.- EXT. MANTISSA) := $M_A \times M_Q$
$E_A + E_Q - 128 > 255$?
Y
N
$E + E - 128 < \phi$?
Y
N
SET CR 15
SET CR 14
(EXPONENT) := $E_A + E_Q - 128$
SET VIOL'N. INT.
SET CR 6
STANDARDISE.
BUT MANTISSA IS "DBLE.- EXT. MANTISSA"
(AB) := MANTISSA & EXPONENT
CNI

NOTE "DBLE.- EXT. MANTISSA" —
DOUBLE - EXTENDED MANTISSA ; THE [illegible]
WITH A 24- BIT EXTENSION TO THE LEFT
SIGNIFICANT END.


DIVIDE
$M_Q = \phi$?
N
Y
$|M_A| > |M_Q|$?
N
Y
ARITH SH. R. M_A
INC E_A
HAS E_A OVERFLOWED ?
N
Y
(MANTISSA) := $M_A \div M_Q$
$E_A - E_Q + 128 > 255$?
N
Y
SET CR 14 + 15
$E_A - E_Q + 128 < \phi$?
N
Y
SET CR 15
SET CR 14
(EXPONENT) := $E_A - E_Q + 128$
STANDARDISE
SET VIOL'N. INT.
SET CR 6
(AB) := MANTISSA & EXPONENT
CNI

Appendix B

Instruction Times for Floating-Point Functions.

The addressing format taken for the Group 1 functions is (s) + D. For other formats, an appropriate adjustment must be made. A store Processor Access Time of 625nS is taken.

8.1. ADDF

The instruction time is given by one of two formulae, depending upon the result of adding the mantissae after the numbers have been adjusted to have the same exponent:-

if result of addition is non-fractional

$$T = (45 + 2n + m) \times \tfrac{1}{8} \quad \mu S$$

if result of addition is fractional

$$T = (44 + 2n) \times \tfrac{1}{8} \quad \mu S$$

where	n =	number of shifts to adjust for same exponent.
	m =	" " " " standardise result.

An average instruction time is 9 μS

8.2. SUBF

The instruction time is as for ADDF, plus $\frac{1}{2}$ μS for the initial negation of the subtrahend.

An average instruction time is $9\frac{1}{2}$ μS

8.3. MLTF

The instruction time is given by

$$T = (66 + 2N + m) \times \tfrac{1}{8} \quad \mu S$$

where N = number of bits set in M_A

In addition, $\tfrac{1}{8}$ μS is added if M_A is negative

An average instruction time is 11 μS

8.4. DIVF

The instruction may be either $15\tfrac{3}{8}$ μS or $15\tfrac{5}{8}$ μS, depending upon conditions at a certain point in the mantissa dividing routine.

An average instruction time is $15\tfrac{1}{2}$ μS

8.5. STND

The instruction time is given by

$$T = (19 + m) \times \tfrac{1}{8} \quad \mu S$$

or if the mantiss is zero, $T = 2\tfrac{1}{4}$ uS

8.6. NEGF

The instruction time is either

$2\tfrac{3}{4}$ μS if the mantissa is zero or -1 ,

or 3 μS if the mantissa is $+\tfrac{1}{2}$

or $2\tfrac{7}{8}$ μS otherwise

An average instruction time is $2\tfrac{7}{8}$ μS

8.7. FIXI or FIXF

The instruction time is given by

$$T = (20 + 2p) \times \tfrac{1}{8}\ \mu S$$

where $p = 31 - (\text{exponent})$ for FIXI

$p = 0 - (\text{exponent})$ for FIXF

An average instruction time is $6\frac{1}{4}$ μS

8.8. FLTI or FLTF

The instruction time is given by

$$T = (18 + q) \times \tfrac{1}{8}\ \mu S$$

where q = the number of successive identical bits, beginning from the most significant end of E

except if $E = \emptyset$, when $T = 1\frac{3}{4}$ uS

An average instruction time is $4\frac{1}{4}$ uS